模块

模块化可以简化大型项目的开发。

一、模块化

模块是Nodejs程序的基本组成部分，文件和模块是一一对应的，一个nodejs文件就是一个模块，这个文件可能是js代码、json或者c c++扩展。

格式有两个：

- CommonJs.
- Es6

1.1 Commonjs

1.1.1 模块引用

var math = require("math")

用require方法引入一个math模块，赋值给变量math，实际上变量名和引入的模块名不必相同，也可以赋值给Math，那后续只支持Math，不支持math，因为不认识。

require方法的参数传入

- 只有字符串，没有路径。引入当前模块下的同级node_modules文件夹下的math模块。如果在当前目录中没有node_modules又或者node_modules文件夹下面没有math模块，则会报错
- 相对路径或绝对路径。

  ```
  var sum = require("./sum");
  ```

  表示引入当前文件的同级目录中的sum.js模块。

1.1.2 定义

module对象。每个模块中，module对象代表该模块本身。
export属性。module对象的一个属性，向外提供接口。

```
//sum.js
function sum(num1, num2){
    return num1 + num2;
}


//main.js
var sum = require("./sum.js");

sum.sum(1 , 2);
```

虽然main中引入了sum，但是无法使用它的sum方法。sum中的sum方法如果想被其他模块使用就需要对外暴露一个接口，export属性用于完成这个工作。

```
//sum.js
function sum(num1, num2){
    return num1 + num2;
}

module.exports.sum =sum;


//main.js
var sum = require("./sum.js");

var result = sum.sum(1 , 2);
```

如此便会成功。

1.1.3 模块标识

模块标识指传给require方法的参数。必须是小驼峰命名的字符串或者再加上相对和绝对路径，如果引入的是js文件，可省略后缀。

1.2 Es6

commonjs的这种模块机制，只适用于服务器端，浏览器端无法适用。

1.2.1 基本导出

export关键字，将已发布的代码部分公开给其它模块。最简单的用法,将export放置在任意变量、函数、或者类声明之前。

```
//导出数据
export var color = "red";

//导出函数

export function sum(num1,num2){
    return num1 + num2;
}

//导出类

export class Rectangle{
    constructor(length ,width){
        this.width = width;
        this.length = length;
    }
}

//定义一个函数，并导出一个函数引用
function multiply(num1,num2){
    return num1 * num2;
}
export {multiply};
//没有在定义的时候倍导出，随后导出引用的方式导出。
```

1.2.2 基本导入

> import {identifier , identifier2} from "./example.js";

- 需要导入的标识符
- 需要导入的标识符的来源模块

1.2.3 重命名的导入

可以在导出的时候重命名。

> as关键字

```
function sum(){

}

export {sum as add};

//sum是本地名称，add是导出名称，这表明另一个模块引入这个函数的时候，必须使用add，不能修改用sum
```

同样可以在导入的时候重命名，as

```
import {add as sum} from "./sum.js";
```

1.3 Es6和Commmjs

1.3.1 CommonJs

- 基本数据类型，复制，数据会被模块缓存，同时另一个模块可以对该模块输出的变量重新赋值
- 复杂数据类型，浅拷贝。对复制出来的对象修改其引用的内容，则原始的也会修改
- require加载某个模块的时候，运行整个模块的代码
- require加载同一个模块的时候，不会再执行，而是取缓存。无论被加载多少次，都只会在最初的一次加载时运行，以后就会加载这个缓存，除非手动清空缓存
- 循环加载，属于加载时执行。脚本代码在require的时候全部执行，一旦出现循环加载，则只输出已经执行的部分，不输出还未执行的部分

1.3.2 Es6

Es6模块中的值属于动态只读引用。

- 只读。不允许修改引入的变量的值，import的变量是只读的，不论是基本还是复杂数据类型。
- 动态。动态是指如果原始值发生变化，import加载的也会变化，不论是基本还是复杂
- 循环加载时，Es6是模块是动态引用，只要两个模块之间存在某个引用，代码就可以执行

1.4 Nodejs的模块

1. NodeJs自身的模块，又叫核心模块，如fs、http等。
2. 用户编写的模块，又叫文件模块。

核心模块在node运行时自动加载，速度最快。
文件模块在运行时动态加载，需要路径分析、文件定位、编译执行，加载比核心模块慢。

模块在加载时，先查询缓存，缓存中没有在找内置模块。如果核心模块没有，则去文件模块中找。

> 缓存模块>核心模块> 用户自定义模块

Nodejs加载无文件类型的优先级是:

> .js > .json > .node

二、NPM

2.1 npm安装

> npm install 模块名

安装好后，包放在工程目录下的node_modules文件夹下，

2.2 全局安装和本地安装

如果当成命令行使用，则全局安装。
如果当成依赖，则本地安装。

> npm install aModule
> npm install bModule -g

2.3 查看安装信息

> npm list -g

查看全局安装的模块。

如果查看某个模块的信息：

> npm list -g aModule

2.4 卸载

> npm uninstall aModule
> npm ls 查看

2.5 更新

> npm update aModule

2.6 搜索

> npm search bModule

2.7 创建模块

> npm init
> npm adduser
> npm publish

四、常见模块

buffer、events、fs、http、net、
path：文件或目录处理
timer：定时器
tls：基于openssl的tls和ssl
dgram： udp